Enhancing learning through self-explanation

Amali Weerasinghe and Antonija Mitrovic

Intelligent Computer Tutoring Group
Department of Computer Science, University of Canterbury
Private Bag 4800, Christchurch, New Zealand
{acw51, tanja}@cosc.canterbury.ac.nz

Abstract: Self-explanation is an effective teaching/learning strategy that has been used in several intelligent tutoring systems in the domains of Mathematics and Physics to facilitate deep learning. Since all these domains are well structured, the instructional material to self-explain can be clearly defined. We are interested in investigating whether self-explanation can be used in an open-ended domain. For this purpose, we enhanced KERMIT, an intelligent tutoring system that teaches conceptual database design. The resulting system, KERMIT-SE, supports self-explanation by engaging students in tutorial dialogues when their solutions are erroneous. We plan to conduct an evaluation in July 2002, to test the hypothesis that students will learn better with KERMIT-SE than without self-explanation.

1. Introduction

Many Intelligent Tutoring Systems (ITS), which provide problem-solving environments, have shown significant learning gains for students particularly in the domain of Mathematics [8] and Physics [9, 16] and Computer Science [10,12]. However, some empirical studies indicate that students tend to keep guessing until they find an action that obtains positive feedback from the ITS [1]. As a result, the student will have difficulties in transferring knowledge to novel situations, even though they gain enough knowledge to obtain a passing grade on tests. Researchers are therefore interested in finding methods that overcome the shallow learning problem. Self-explanation is described as an "*activity of explaining to oneself in an attempt to make sense of new information, either presented in a text or in some other medium*" [4], and has been shown to facilitate the acquisition of deep knowledge [5]. Since explaining the instructional material to oneself facilitates the integration of new information into existing knowledge, self-explanation can be viewed as a knowledge construction activity [4]. However, the results of Chi's study [4] indicated that self-explanation is not merely a process of generating inferences to fill gaps in knowledge, but a process of repairing one's own mental model of the topic under study. In this context, self-explanation facilitates the identification and removal of misconceptions. Therefore, self-explanation also promotes reflection, which is a meta-cognitive skill many students lack [13].

KERMIT (Knowledge-based Entity Relationship Modelling Intelligent Tutor) [12, 15] is a problem-solving environment that supports students learning database (DB) modelling. In this paper, we describe how we enhanced KERMIT to support self-explanation. Section 2 describes related work. KERMIT is briefly introduced in Section 3 and KERMIT-SE, its enhancement that facilitates self-explanation, is presented in the next section. The conclusions and directions for future research are given in the final section.

2. Related Work

There are only a few systems that support self-explanation. SE-Coach facilitates self-explanation by prompting students to explain example solutions [7]. It is implemented within ANDES [9,16] a tutoring system that teaches Newtonian Physics. The first level of scaffolding in the SE-Coach's interface is provided by a masking mechanism that covers different parts of the example with grey boxes, each corresponding to a unit of information. When the student moves the mouse over a box, it disappears, revealing the content underneath. The second level of scaffolding produces specific prompts to self-explain. Whenever a student unmasks a piece of an example, s/he is prompted to self-explain if a concept worthy of explanation is uncovered. The students are expected to self-explain not only principles of the domain but also solution steps. However, the students are prompted to self-explain only when the tutor decides it is beneficial. To determine when to intervene, SE-Coach relies on a probabilistic student model, that monitors how well the student understands the domain by capturing both implicit self-explanations and self-explanations generated through the interface [6]. The results of the empirical evaluation of SE-Coach reveals that the structured scaffolding of self-explanation can be more beneficial in the early learning stages. It also suggests that even simpler forms of prompting can successfully trigger self-explanation when students become more proficient in the subject matter.

ANDES has been further enhanced by incorporating ATLAS [17], a module to conduct natural language dialogues to facilitate deep learning. When ATLAS recognizes an opportunity to encourage deep learning, it initiates a natural language dialogue with the student. The main objective of the dialogues is to facilitate

knowledge construction; hence, these dialogues are known as knowledge construction dialogues (KCDs). KCDs provided by ATLAS are currently limited to teaching domain principles. A limited study (with ten participants) revealed that the students interacting with ATLAS learnt significantly better than students who interacted with ANDES [11]. However, larger studies are needed to obtain more reliable results.

Using the PACT Geometry Tutor, Aleven and Koedigner [1] investigated how self-explanation facilitates deep learning. The students in the experimental group were expected to provide correct explanations for solution steps by citing definitions and theorems used. A glossary of knowledge in the form of definitions and theorems was provided in order to help students to explain the solution steps. The study revealed that explaining the reasoning steps results in improved problem solving skills. The researchers also discovered that students who explained solution steps attempted significantly fewer problems than their colleagues who provided only the answers, although both groups spent the same amount of time with the tutor. However, there was evidence that students who explained the solutions required fewer problems to reach the tutor's mastery level criterion.

PACT Geometry Tutor has been further enhanced to facilitate self-explanation through natural dialogue [2]. Analysis of several corpora revealed that it was difficult for students to state geometry theorems in their own words. The analysis further indicated that there are many ways to state a theorem correctly and even more ways to state it incorrectly. Therefore, the system uses a "classify-and-react" approach to develop the tutor's interactions. i.e. in each dialogue cycle, the tutor classifies the student's explanation and then responds based on the classification. The system is currently being developed and an evaluation is yet to be conducted.

AUTOTUTOR [10] is used in an introductory course in computer literacy. The system improved the students' learning by 0.5 standard deviation units when compared with a control group of students who read the same chapters from a book. AUTOTUTOR requires students to provide lengthy explanations for the *How*, *Why* and *What-if* type of questions. This approach encourages students to articulate lengthier answers that exhibit deeper reasoning instead of short answers, which may lead to shallow knowledge. A continuous multi-turn dialogue between the tutor and the student takes place throughout the session.

These systems use different approaches to facilitate self-explanation, depending on the domain and the target student group. Problem solving activities in these domains are well structured, and the types of self-explanations expected from students can be clearly defined. For example, in Mathematics and Physics, students are expected to explain the theorems that they have used. In computer literacy, the students are expected to explain the definitions, meaning of terms and how a certain task is being carried out. However, it is challenging to incorporate self-explanation in the domain of database design, as it is an open-ended task: there is an outcome defined in abstract terms, but there is no procedure to use to find that outcome. It is not sufficient to ask the students to explain the concepts of database modelling, as the database design skills can only be developed through extensive practice.

3. K*ER*MIT: A Knowledge-based ER Modelling Tutor

KERMIT (Knowledge-based Entity Relationship Modelling Intelligent Tutor) is an ITS aimed at the university-level students learning conceptual database design. The architecture of the system is illustrated in Figure 1. For a detailed discussion of the system, see [15]; here we present some of its basic features. KERMIT is a problem-solving environment in which students can practice database design using the Entity Relationship (ER) data model. The system is intended to complement traditional instruction, and assumes that students are familiar with the ER model. The system consists of an interface, a pedagogical module, which determines the timing and content of pedagogical actions, and a constraint-based modeller, which analyses student answers and generates student models.

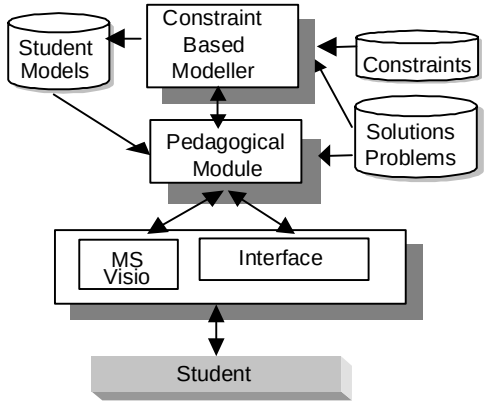

Fig. 1. The architecture of KERMIT

KERMIT contains a set of problems and the ideal solutions to them, but has no problem solver. In order to check the correctness of the student's solution, KERMIT compares it to the correct solution, using domain knowledge represented in the form of more than 90 constraints. It uses Constraint-Based Modelling [14] to model the domain and student's knowledge. The constraints cover both syntactic and semantic knowledge. The syntactic constraints are concerned with syntactic details in a student's solution. An example of such a constraint is *"An entity cannot be directly connected to another entity*". Semantic constraints relate the student's solution to the system's ideal solution. For example, there are constraints that check for equivalent, but not identical, ways of modelling parts of a database in the student's and ideal solution. One such constraint deals with multi-valued attributes of entities, which may alternatively be modelled as weak entities. Students have several ways of selecting problems in KERMIT. They may work their way through a series of problems, arranged according to their complexity. The other option is a

system-selected problem, when the pedagogical module selects a problem for the student on the basis of his/her student model. The interface is composed of three windows tiled horizontally. The top window displays the current problem and provides controls for stepping between problems, submitting a solution and selecting feedback level. The middle window is the main working area. In this window the student draws ER diagrams using the toolbar on the left side of the window. Feedback is presented in the lowest window in the textual form, as well as through an animated pedagogical agent.

4. Design and Implementation

All systems that facilitate self-explanation prompt students to explain most of the problem-solving steps, requiring students to point out the definitions/theorems used. We believe this approach is not sufficient to acquire robust knowledge. Therefore, our tutor prompts for self-explanation only when the student violates a constraint, which indicates missing/erroneous knowledge, or a slip. The tutor is thus able to customise self-explanation based on the student model so that the knowledge construction is facilitated for students who have misconceptions or gaps in their knowledge without disrupting others [3].

Since a student can make several errors at each submission, the pedagogical module (PM) needs to decide on which error to initiate the self-explanation process. The constraints in the knowledge base are ordered according to the traditional ER modelling procedure, starting with modelling entities first, then relationships and finally the attributes. This ordering alone is not sufficient to select an error to prompt for self-explanation, as most errors violate more than one constraint. At the same time, semantic constraints are not specific enough to guide self-explanation effectively. For instance, the constraint "*A construct that should be a regular entity has been represented by another type*" is violated when a regular entity is represented either as a simple attribute or a weak entity. Different approaches need to be taken in these two cases. Self-explanation in the first case needs to help the student to clarify the definitions of entities and attributes so that the student understands when to use them. In the latter case, the student should understand the differences between regular and weak entities, which will enable the error to be corrected and the correct design decisions made in future. Also, the PM should enable the students to build a more comprehensive mental model of the domain knowledge by giving them an opportunity to learn basic concepts before complicated ones.

We have analysed different students' errors and arranged them into a hierarchy. Nodes in this hierarchy are ordered from basic domain principles to more complicated ones. Violated constraints for each type of error are represented as leaves of the hierarchy. Self-explanation is facilitated through tutorial dialogue. We designed a tutorial dialogue for each type of error. When the student submits a solution, the student modeller evaluates it against the constraint base and identifies the violated constraints. The pedagogical module then searches for the first tutorial dialogue that covers the same violated constraints. Since the dialogues are ordered according to the complexity of the domain principles that the student needs to learn, PM selects the dialogue by traversing the hierarchy in a top-to-bottom, left-to-right manner, selecting the first dialogue that involves some or all violated constraints. The chosen dialogue is displayed in the feedback window.

For example, consider the following problem statement: "*For each course a student has taken, we need to know the final grade. Each course has a unique course code and a student has his/her student id.*" Figure 1 shows a student's solution and the ideal solution. The student's solution contains several errors: the *HAS* relationship is not needed, the *GRADE* entity should be represented as an attribute, and also it is not possible to have en entity without any attributes. The pedagogical module selects the dialogue corresponding to the simplest error (*GRADE* represented as an entity instead of as an attribute) to start with. Figure 2 presents a sample dialogue, which may occur between the student and the tutor. In the first level of a dialogue, the student is informed that the action s/he has performed is incorrect and is asked to interpret their action in the context of ER modelling (*tutor-1*). A list of possible answers is provided from which the correct one can be selected. If the student fails to provide the correct answer or indicates that s/he needs more help (*student-1*), s/he will be asked more specific questions that provide a further opportunity to understand the fundamental principle that is violated (*tutor-2*). However, if s/he fails to identify the mistake and correct it, even after going through a series of detailed questions, as a last resort the tutor will specify the correct action to be performed together with a brief description about the fundamental principle that needs to be learnt (*tutor-7, tutor-8*). KERMIT-SE uses various types of interactions such as simple questions (*tutor-1*), fill-in-a-blank (*tutor-8*), or true-false questions, to motivate the student to self-explain. Since all dialogues initiate self-explanation by making the student aware of the incorrect action (*tutor-1*), students are given an opportunity to reflect on their problem solving procedure, which is another important meta-cognitive skill.

If the student violates the same constraint in a subsequent attempt, a choice is given to initiate the tutor's behaviour. The student can decide (i) to go through the tutorial dialogue for that constraint again, (ii) to request a single question which will enable an understanding of the fundamental principle violated or (iii) attempt to correct the mistake. This way, the student is given greater control and increased motivation to use the system. It will also provide an opportunity to identify students' self-assessment skills [13].

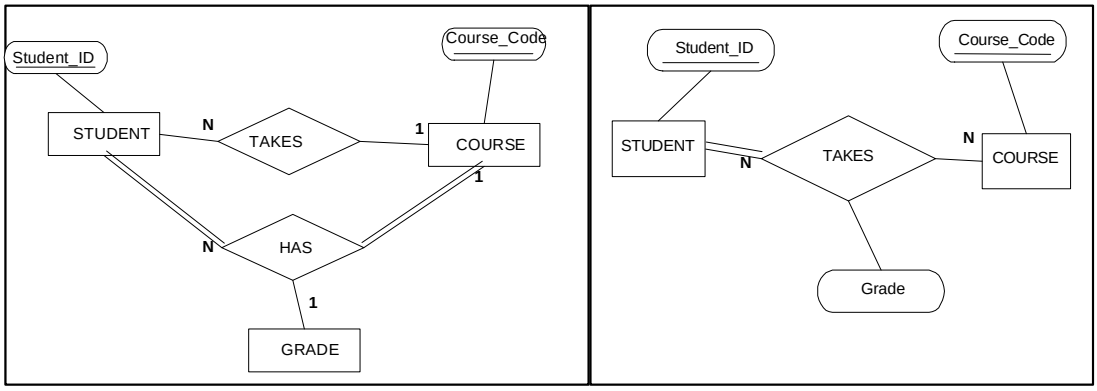

Figure 1: A student's solution and the correct solution

Tutor-1: You've modelled *Grade* as an entity. What does that mean in terms of properties of an entity
Student-1: I don't know
Tutor-2: Let's try a simple thing. How do we model characteristics of an entity ?
Student-2 : Using Attributes
Tutor-3 : Great ! Now tell me why modelling *Grade* as an entity contradicts with the problem description ?
Student-3 : I'm not sure
Tutor-4: Don't worry. I'm here to help you. Check carefully whether you can find any attributes for *Grade*.
{After sometime} Pls. tell me whether you manage to find any attributes for *Grade*
Student-4: No
Tutor-5: Fine :It doesn't have any attributes. So now can you tell me why modelling *Grade* as an entity contradicts with the problem description ?
Student -5: Because it doesn't have any attributes to describe its characteristics
Tutor-6: Great! Now tell me the best way to model *Grade*?
Student-6 : I need more help
Tutor-7: Ok. If *Grade* doesn't have attributes, it cannot be an entity. So it should either be a relationship or an attribute. Now tell me the best way to model *Grade*.
Student-7 : Attribute
Tutor-8 : Great ! Now use the correct word to fill in the blanks so that you'll remember what you learnt.
…….. are used to describe characteristics of an …….. in a ER model
Student-8 : attributes, entity
Tutor-9: Excellent, Go ahead and do the changes. Submit the solution when you're done

Figure 2: A sample tutorial dialogue

The short-term student model in KERMIT consists of a list of satisfied and a list of violated constraints for the student's last submission, while the long-term model records the history of each constraint (how often a constraint has been relevant, and how often it has been satisfied/violated) [15]. In KERMIT-SE, the long-term model additionally records the type of error made and the level of prompting the student needed to correct his/her mistake for every violated constraint.

5. Conclusions and Future Work

Self-explanation is a way of supporting deep learning. This research focuses on incorporating self-explanation into a tutor that teaches the open-ended task of ER modelling. KERMIT-SE supports self-explanation by engaging students in tutorial dialogues about errors they make. We analysed typical student errors, and designed a tutorial dialogue for each type of error. These dialogues are arranged in a hierarchy, according to the complexity of the domain concepts they include. Each error corresponds to one or more violated constraints. The pedagogical

module selects the simplest dialogue based on the student's solution. The student model is also modified to represent the student's skills at explaining their problem-solving decisions.

An evaluation study is planned for July 2002. The study aims to discover whether guided self-explanation will help students to learn with understanding in the domain of database modelling. It will involve second-year university students enrolled in an introductory database course. We are interested in investigating whether the experimental group (who will use KERMIT-SE) will learn significantly better compared to the control group (who will use KERMIT). We are also interested in assessing whether there is a correlation between the level of prompting required by the students to identify their mistake and their prior knowledge. Since the student model records the details of types of errors associated with every violated constraint, the student model can be used to identify the concepts that are difficult for each student. We are interested in assessing whether there is a correlation between the level of prompting needed by each student and their perceived level of difficulty of the principle they violated. The data recorded in the student model will be further analyzed to ascertain whether students gain the ability to identify their mistakes through self-explanation, with very little prompting, when they become more proficient in the subject matter. Since KERMIT-SE gives the learner more control to initiate the tutor's behaviour, the evaluation study will provide us with an opportunity to understand how well the students can use self-assessment to acquire robust domain knowledge. Also, the empirical results will indicate whether the pedagogical module needs to be enhanced to provide more assistance to the students who lack self-assessment skills in initiating the tutor's behaviour and selecting new problems. Since self-explanation has not been previously used in a design environment to foster learning, this research will provide important information.

Acknowledgements

We thank Pramuditha Suraweera and Danita Hartley for their help in implementing KERMIT-SE. This research was made possible by the NZODA postgraduate scholarship awarded to the first author.

References

1. Aleven, V., Koedinger, K. R. and Cross, K. Tutoring Answer Explanation Fosters Learning with Understanding. In: Artificial Intelligence in Education, Lajoie, S.P. and Vivet, M.(eds.), Amsterdam : IOS Press (1999) 199-206.
2. Aleven, V., Popescu, O. and Koedinger, K. R. Towards Tutorial Dialogue to Support Self-Explanation: Adding Natural Language Understanding to a Cognitive Tutor. Int. Journal on Artificial Intelligence in Education, 12 (2001) 246-255.
3. Bunt, A. and Conati, C. Modeling Exploratory Behaviour. In: Bauer, M., Gmytrasiewicz, P. J. and Vassileva, J. (eds.), Proc. of 8th International Conference, User Modeling, Sonthofen, Germany (2001) 219-221
4. Chi, M. T. H. Self-explaining Expository Texts: The dual processes of generating inferences and repairing mental models. Advances in Instructional Psychology, (2000) 161-238.
5. Chi, M. T. H., Bassok, M., Lewis, W., Reimann, P. and Glaser, R. Self-Explanations: How Students Study and Use Examples in Learning to Solve Problems. Cognitive Science, 13 (1989) 145-182.
6. Conati, C. and VanLehn K., Providing Adaptive Support to the Understanding of Instructional Material. In Proc. IUI 2001 Sante Fe, New Mexico (2001).
7. Conati, C. and VanLehn, K. Toward Computer-Based Support of Meta-Cognitive Skills: a Computational Framework to Coach Self-Explanation. Int. J. Artificial Intelligence in Education, 11 (2000) 389-415.
8. Corbett, A.T.M., Trask, H.J., Scarpinatto, K.C. and Handley, W.S. A formative evaluation of the PACT Algebra II Tutor : support for simple hierarchical reasoning. Proc. ITS'98 374-383.
9. Gertner A. S. and VanLehn, K. ANDES: A Coached Problem-Solving Environment for Physics, In Gauthier G., Frasson, C. and VanLehn, K. (eds.), Proc. ITS 2000, Montreal (2000) New York : Springer 133-142.
10. Grasser, A. C., Wiemer-Hastings, K. Wiemer-Hastings, P. and Kreuz, R., Tutoring Research Group 1999. AUTOTUTOR: A Simulation of a Human Tutor. Journal of Cognitive Systems Research 1(1) (1999) 35-51.
11. Grasser, A.C., VanLehn, K., Rose, C.P., Jordan, P.W. and Harter, D. Intelligent Tutoring Systems with Conversational Dialogue, AI Magazine, American Association for Artificial Intelligence, Winter 2001, 39 -51
12. Mitrovic, A., Mayo, M., Suraweera, P. and Martin, B. Constraint-based tutors: a success story. In Monostori, L., Vancza, J. and Ali, M. (eds.), Proc. of IEA / AIE, Budapest, Hungary 2001, Springe-Verlag 931-940.
13. Mitrovic, A. Investigating Students' Self-assessment Skills. In Bauer, M., Gmytrasiewicz, P. J. and Vassileva, J. (eds.), Proc. UM 2001, Berlin Heidelberg (2001) Springer-Verlag 247-250.
14. Ohlsson, S. Constraint-based Student Modelling. In: Greer, J.E., McCalla, G (eds) Proc. of Student Modelling: the Key to Individualized Knowledge-based Instruction, Springer-Verlag Berlin (1994) 167-189.
15 Suraweera, P. and Mitrovic, A. KERMIT: a constraint-based tutor for database modelling. Proc. ITS'2002 (in press).
16. VanLehn, K. Conceptual and Meta-learning during Coached Problem Solving. In: Frasson, C., Gauthier, G. and Lesgold, A. (eds.), Proc ITS 1996 Berlin (1996) 29-47.
17. VanLehn, K., Freedman, R., Jordan, P., Murray, C., Osan, R., Ringenberg, M., Rose, C.P., Schulze, K., Shelby, R., Treacy, D., Weinstein, A. and Wintersgill, M. Fading and Deepening: The Nest steps for ANDES and Other Model-Tracing Tutors. In Proc. ITS 2000, Gauthier, G., Frasson, C. and VanLehn, K. (eds.), Montreal (2000) 474-483.